UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/976,930	10/11/2001	Clifford L. Hersh	PA1950US	2048

22830 7590 01/12/2006

CARR & FERRELL LLP
2200 GENG ROAD
PALO ALTO, CA 94303

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2195	

DATE MAILED: 01/12/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/976,930	HERSH, CLIFFORD L.	
	Examiner	Art Unit	
	Lewis A. Bullock, Jr.	2195	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 August 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *11 October 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☒ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *10/27/05*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1-5, 7-10, 13-20 and 22-40 are rejected under 35 U.S.C. 102(b) as being anticipated by "Managing Long Linked Lists Using Lock-Free Techniques" by FAROOK et al.

As to claim 1, FAROOK teaches a method for executing on an operation upon a linked data structure (linked list) having at least one element (list node) (pg. 5, Non-blocking Linked list), the method comprising the steps of: performing a first set of operation tasks in a first phase, the first set of operation tasks operable to effect a first set of element state transitions (via performing the cursor operation to find the correct position in the list for insertion or deletion / via creating the new node, pointing its next identifier to the c.target and checking if c.target has been changed) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation); developing a second set of operation tasks, the second set of operation tasks operable to effect a second set of element state transitions (via determining if the cursor context has changed in order to retry the operation or continue performing the operation by setting the prev.next to refer to the new node or delete the node), the second set of

element state transitions being distinct from the first set of element state transitions; and performing the second set of operation tasks in a second phase (via inserting the node in the list or deleting the node from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 10, FAROOK teaches a method for performing insertion and deletion operations (insert and delete) on elements in a linked data structure (list nodes on a linked list) (pg. 5, Non-blocking Linked list), the method comprising the steps of: performing a first set of operation tasks in a first phase for each insertion and deletion operation, the first set of operation tasks operable to effect a first set of element state transitions (via performing the cursor operation to find the correct position in the list for insertion or deletion / via creating the new node, pointing its next identifier to the c.target and checking if c.target has been changed) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation); developing a second set of operation tasks for each insertion and deletion operation, the second set of operation tasks operable to effect a second set of element state transitions (via determining if the cursor context has changed in order to retry the operation or continue performing the operation by setting the prev.next to refer to the new node or delete the node), the second set of element state transitions being distinct from the first set of element state transitions; and performing the second set of operation tasks in a second phase (via

inserting the node in the list or deleting the node from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 13, FAROOK teaches a method of inserting an element into a linked data structure (list nodes of a linked list) (pg. 5, Non-blocking Linked list) comprising the steps of: performing a first set of operating tasks in a first phase, the first set of operation tasks operable to effect a first set of element state transitions, including a pre-associated state to a pending insert state transition (via performing the cursor operation to find the correct position in the list for insertion or deletion / via creating the new node, pointing its next identifier to the c.target and checking if c.target has been changed) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation); developing a second set of operation tasks, the second set of operation tasks operable to effect a second set of element state transitions including a pending insert state to a valid state transition (via determining if the cursor context has changed in order to retry the operation or continue performing the operation by setting the prev.next to refer to the new node or delete the node); and performing the second set of operation tasks in a second phase (via inserting the node in the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 17, FAROOK teaches a method of deleting an element from a linked data structure comprising the steps of: performing a first set of operation tasks in a first phase, the first set of operation tasks operable to effect a first set of element state transitions including a valid state to a pending delete state transition (via performing the cursor operation to find the location of target node (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 11-12, wherein the enhancement to the prior teachings teach the limitation); developing a second set of operation tasks, the second set of operation tasks operable to effect a second set of element state transitions including a pending delete state to an invalid state transition (via performing the TryDelete operation which determines whether the deletion can be performed based on the state of the affected nodes) (pg. 11-12); and performing the second set of operation tasks in a second phase (via deleting the node from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 22, FAROOK teaches a method for executing an operation upon a linked data structure having at least one element (list nodes of a linked list) (pg. 5, Non-blocking Linked list), the method comprising the steps of: creating first and second sets of operation tasks, the first set of operation tasks being characterized by navigation of the linked data structure using at least an existing link (via performing the cursor algorithm), and the second set of operation tasks being distinct from the first set of

operation tasks and being characterized by at least a pointer to the linked data structure (via performing an insert or delete operation based on the cursor position); and performing the first set of operation tasks in a first phase and the second set of operation tasks in a second phase (via inserting the node in the list or deleting the node from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 23, FAROOK teaches a method for executing an operation upon a linked data structure having at least one element (list nodes of a linked list) (pg. 5, Non-blocking Linked list), the method comprising the steps of: dividing the operation into first and second distinct sets of operation tasks (via the insertion and deletion algorithm comprising a cursor algorithm for finding the correct position and an actually insertion or deletion algorithm for performing the operation at that position); performing the first set of operation tasks in a first phase (cursor function); and performing the second set of operation task in a second phase (via performing the TryInsert / Try delete operation) (pgs 9-14).

As to claim 36, FAROOK teaches a consistent method of executing simultaneous operations (concurrent insert and delete operations) (pg. 14, Theorems 1-3) on a linked data structure having at least one element (list nodes on a linked list) (pg. 5, Non-blocking Linked list), the method comprising the steps of: performing any first phase

operation task of each of the simultaneous operations in a first phase using parallel processes (processes that are trying to performing operations on adjacent nodes) (via performing the cursor operation to find the correct position in the list for insertion or deletion / via creating the new node, pointing its next identifier to the c.target and checking if c.target has been changed) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation); developing a set of serial operations during the first phase (via only one process succeeds while the others fail) (pg. 14, theorems 1-3); and performing any second operation tasks of each of the simultaneous operations in a second phase, the second phase operation task including at least one of the set of serial operations (via inserting the node in the list or deleting the node from the list by the succeeded process) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation)..

As to claims 28 and 29, reference is made to a system that corresponds to the method of claims 1 and 23 and is therefore met by the rejection of claims 1 and 23 above.

As to claims 30 and 31, reference is made to a computer readable medium that corresponds to the method of claims 1 and 23 and is therefore met by the rejection of claims 1 and 23 above.

As to claims 32 and 33, reference is made to a system that corresponds to the method of claims 1 and 23 and is therefore met by the rejection of claims 1 and 23 above.

As to claims 34 and 35, reference to claim 22 for rejection.

As to claim 2, FAROOK teaches the first set of operation tasks includes navigating existing data structure links (via the cursor function) (pg. 9-10, linked list traversal).

As to claim 3, FAROOK teaches developing pointers to the data structure (via determining if the cursor context has changed in order to retry the operation or continue performing the operation by setting the prev.next to refer to the new node or delete the node), the pointers being used in the step of performing the second set of operation tasks in a second phase (via inserting the node in the list or deleting the node from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 4, FAROOK teaches the operation tasks of the second set of operation tasks are performed atomically (via inserting the node in the list or deleting

the node from the list using CAS or DCAS instructions) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 5, FAROOK teaches developing the second set of operation tasks as a list (via inserting the node in the list or deleting the node from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 7, FAROOK teaches a step of performing a conflicts check for the operation (via if the node is being accessed by another process) (via the try_* operations) (pg. 11-13).

As to claim 8, FAROOK teaches the first set of element state transitions comprises: a valid state to a pending delete state transition; a pre-associated state to a pending insert state transition; and a pending insert state to a hidden state transition (via performing the cursor function in order to get to the correct insert or delete positions for performing the final state, actually insertion or deletion into/from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 9, FAROOK teaches the second set of element state transitions comprises: a pending insert state to a valid state transition; a pending delete state to a invalid state transition; a hidden state to a invalid state transition; a pending delete state to a valid state transition; a hidden state to a pending insert state transition; and a pending insert state to an invalid state transition (via performing the TryDelete or TryInsert function in order to performing the operation and reach the final state, actually insertion or deletion into/from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 14, FAROOK teaches the pre-associated state to a pending insert state transition is accomplished by: marking the element to be inserted as being pre-associated to the data structure (via creating the new node wherein it has the same features as the other nodes (pg. 9, "each node in the linked list consists of four fields..."); navigating the data structure to an insertion point (via the cursor algorithm); creating links between the element to be inserted and the data structure at the insertion point (via pointing its next identifier to the c.target and checking if c.target has been changed) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation), the links created being visible only to the insertion operation (via the traverse pointer is not changed) (pg. 9); and marking the element as being pending insert (via the counter field has not changed) (pg. 9).

As to claim 15, FAROOK teaches the pending insert state to a valid state transition is accomplished by: creating instructions for making the created links visible to all operations; and creating instructions for making existing links at the insertion point invisible to all operations (via performing the TryInsert operation to attempt to perform in the insert and thereby make the nodes links visible) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 16, FAROOK teaches executing the created instructions including marking the element as valid (via returning true in the tryInsert operation such that the node is inserted in the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation).

As to claim 18, FAROOK teaches the valid state to a pending delete state transition is accomplished by: navigating the data structure to a deletion point (via the cursor algorithm); creating links at the deletion point visible only to the deletion operation; and marking the element to be deleted as pending delete (via calling TryDelete to attempt to delete the target node by setting the next field of node prev to point to node next) (pg. 11-12).

As to claim 19, FAROOK teaches the pending delete state to an invalid state transition is accomplished by: creating instructions for making the created links visible to all operations; and making existing links at the deletion point invisible to all operations (via updating of the pointers and retrying the deletion of the node by setting the next field of node prev to point to node next) (pg. 11-12).

As to claim 20, FAROOK teaches executing the instructions including marking the element to be deleted as invalid (via indicating Delete_Again and retry the deleting operation / or determining that the counter field is zero and therefore the node cannot be deleted) (pg. 11-12; pg. 9).

As to claim 24, FAROOK teaches the first set of operation tasks is operable to maintain the linked data structure in an existing linked state (via the cursor is operable to traverse the linked list) (pg. 9).

As to claim 25, FAROOK teaches the second set of operation tasks operable to modify the existing linked state (via being able to insert or delete nodes to the linked list) (pg. 11-13).

As to claim 26, FAROOK teaches the first set of operation tasks is visible only to the operation (via the cursor operation is executed initially by the insert and delete operation to find the position of insertion or deletion) (pg. 9-13).

As to claim 27, FAROOK teaches the second set of operation tasks is visible to each of a plurality of operations upon the linked data structure (via actually performing the insertion and deletion such that new processes when traversing the list can view the new node) (pg. 11-13).

As to claim 37, FAROOK teaches at least one of the simultaneous operations includes an element insertion operation, the first phase operation task of the element insertion operation being performed on an unlocked portion of the linked data structure (insert operation) (pg. 12-13).

As to claim 38, FAROOK teaches at least one of the simultaneous operations includes an element deletion operation, the second phase operation task of the element deletion operation being performed independently of navigation of the linked data structure (delete operation) (pg. 12 – 13) (wherein one can still traverse the list based on the traverse pointer) (pg. 9).

As to claim 39, FAROOK teaches the first phase operation tasks are asynchronous and use existing links to navigate the linked data structure (via traversing the list) (pg. 9-13).

As to claim 40, FAROOK teaches the first phase operation tasks of more than one of the simultaneous operations are completed before the second phase of any of the simultaneous operations is initiated (via traversing the list independently of other operations) (pg. 9-13).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 6, 11, 12 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Managing Long Linked Lists Using Lock-Free Techniques" by FAROOK et al.

As to claim 11, FAROOK teaches a method for executing operations upon a linked data structure having at least one element (list nodes of a linked list) (pg. 5, Non-blocking Linked list), the method comprising the steps of: performing a first set of operation tasks in a first phase, the first set of operation tasks operable to effect a first set of element state transitions (via performing the cursor operation to find the correct position in the list for insertion or deletion / via creating the new node, pointing its next identifier to the c.target and checking if c.target has been changed) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation);

developing a second set of operation task, the second set of operation tasks operable to effect a second set of element state transitions (via determining if the cursor context has changed in order to retry the operation or continue performing the operation by setting the prev.next to refer to the new node or delete the node), the second set of element state transitions being distinct from the first set of element state transitions; and performing the second set of operation tasks in a second phase (via inserting the node in the list or deleting the node from the list) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation). However, FAROOK does not teach queuing operation tasks in a task queue and receiving the queued operation tasks. Official Notice is taken in that it is well known in the art that called operations can be stored in a queue and serialized for execution and therefore would be obvious to one skilled in the art that applications can queue and serialize the execution of operations on a linked list.

As to claim 12, FAROOK teaches repeating the steps (repeat loop to re-execute) (pg. 11).

As to claim 21, FAROOK teaches a method for executing an operation upon a linked data structure having at least one element (list nodes of a linked list) (pg. 5, Non-blocking Linked list), the method comprising the steps of: a first set of operation tasks operable to effect a first set of element state transitions; performing the first set of

operation tasks in a first phase (via performing the cursor operation to find the correct position in the list for insertion or deletion / via creating the new node, pointing its next identifier to the c.target and checking if c.target has been changed) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation); a second set of operation tasks operable to effect a second set of element state transitions, the second set of element state transitions being distinct from the first set of element state transitions; and performing the second set of operation tasks in a second phase transitions (via determining if the cursor context has changed in order to retry the operation or continue performing the operation by setting the prev.next to refer to the new node or delete the node and inserting the node in the list or deleting the node from the list based on the determination) (pg. 5-6, 2.3 Non-Blocking Linked list wherein a prior teaching discloses the limitation and on page 12-14 and 11-12, wherein the enhancement to the prior teachings teach the limitation). However, FAROOK does not explicitly detail the grouping of a first plurality of operation tasks to a first set of operations tasks to be performed. FAROOK details a new implementation developed such that certain algorithms, i.e. the cursor, insert, and delete algorithms execute a plurality of instruction operations. Therefore, it would be obvious to the teachings of FAROOK that the development of the various algorithms would require one to group the instruction operations into a set, i.e. algorithm or sub program to be run and executed as detailed in FAROOK.

As to claim 6, FAROOK substantially discloses the invention above. However, FAROOK does not teach that the list is a first in last out list. Official Notice is taken in that a first in last out list, i.e. a list wherein items are added on one end and removed from the other end such that the a node added on the insert end is the last item to be removed from the other end, are well known in the art and therefore would be obvious to one of ordinary skill in the art that the list of FAROOK functions accordingly with the added functions of intermediate insertion and deletion.

Pertinent Prior Art Cited, but not Relied Upon

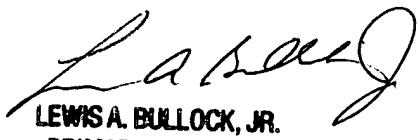
There are various other documents cited in the Notice of References Cited that disclose similar teachings to those of Farook wherein a linked list is manipulated to insert or delete elements such that the state is changed. In particular, Applicant is referred to publication, "A Nonblocking Algorithm for Shared Queues Using Compare-and-Swap" by Prakash et al. This reference details the transition of a queue, linked list, between intermediate states such that insertion or deletion, i.e. enqueue or dequeue operations, are performed. If Applicant believes that the cited prior art has no probative value to the invention, the Examiner requests Applicant to provide some understanding of this in order to avoid having the cited art later relied upon.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 a.m. - 5:00 p.m..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER

January 9, 2006